Improving speech recognition by revising gated recurrent units

Mirco Ravanelli[1], *Philemon Brakel*[2], *Maurizio Omologo*[1], *Yoshua Bengio*[2]

[1]Fondazione Bruno Kessler, Trento, Italy
[2]Université de Montréal, Montréal, Canada

mravanelli@fbk.eu, pbpop3@gmail.com, omologo@fbk.eu, yoshua.umontreal@gmail.com

Abstract

Speech recognition is largely taking advantage of deep learning, showing that substantial benefits can be obtained by modern Recurrent Neural Networks (RNNs). The most popular RNNs are Long Short-Term Memory (LSTMs), which typically reach state-of-the-art performance in many tasks thanks to their ability to learn long-term dependencies and robustness to vanishing gradients. Nevertheless, LSTMs have a rather complex design with three multiplicative gates, that might impair their efficient implementation. An attempt to simplify LSTMs has recently led to Gated Recurrent Units (GRUs), which are based on just two multiplicative gates.

This paper builds on these efforts by further revising GRUs and proposing a simplified architecture potentially more suitable for speech recognition. The contribution of this work is two-fold. First, we suggest to remove the reset gate in the GRU design, resulting in a more efficient single-gate architecture. Second, we propose to replace tanh with ReLU activations in the state update equations. Results show that, in our implementation, the revised architecture reduces the per-epoch training time with more than 30% and consistently improves recognition performance across different tasks, input features, and noisy conditions when compared to a standard GRU.

Index Terms: speech recognition, deep learning, recurrent neural networks, LSTM, GRU

1. Introduction

Building machines that are able to recognize speech represents a fundamental step towards flexible and natural human-machine interfaces. A primary role for improving such a technology is played by deep learning [1], which has recently contributed to significantly outperform previous GMM/HMM speech recognizers [2]. During the last years, deep learning has been rapidly evolving, progressively offering more powerful and robust techniques, including effective regularization methods [3, 4], improved optimization algorithms [5], as well as better network architectures. The early deep learning works in speech recognition were mainly based on standard multilayer perceptrons (MLPs) [6, 7], while recent systems benefit from more advanced architectures, such as Convolutional Neural Networks (CNNs) [8] and Time Delay Neural Network (TDNN) [9, 10].

Nevertheless, since speech is inherently a sequential signal, it would be natural to address it with Recurrent Neural Networks (RNNs), which are potentially able to properly capture long-term dependencies. Several works have already highlighted the effectiveness of RNNs in various speech processing tasks such as speech recognition [11, 12, 13, 14, 15], speech enhancement [16], speech separation [17, 18] as well as speech activity detection [19]. Recent results in the newborn field of end-to-end speech recognition [20, 21] have also shown that RNNs are promising candidates for replacing traditional Hidden Markov Models (HMMs) in DNN/HMM speech recognizers.

Training RNNs, however, can be complicated by vanishing and exploding gradients, which might impair learning long-term dependencies [22]. Although exploding gradients can effectively be tackled with simple clipping strategies [23], the vanishing gradient problem requires special architectures to be properly addressed. A common approach relies on the so-called gated RNNs, whose core idea is to introduce a gating mechanism for better controlling the flow of the information through the various time-steps. Within this family of architectures, vanishing gradient issues are mitigated by creating effective "shortcuts", in which the gradients can bypass multiple temporal steps.

The most popular gated RNNs are Long Short-Term Memory networks (LSTMs) [24], which often achieve state-of-the-art performance in several machine learning tasks. LSTMs rely on a network design consisting of memory cells which are controlled by forget, input, and output gates. Despite their effectiveness, such a sophisticated gating mechanism might result in an overly complex model that can be tricky to implement efficiently. On the other hand, computational efficiency is a crucial issue for RNNs and considerable research efforts have recently been devoted to the development of alternative architectures [25, 26, 27]. With this purpose, an attempt to simplify LSTMs has led to a novel model called Gated Recurrent Unit (GRU) [28, 29], which is based on just two multiplicative gates. Despite the adoption of a simplified gating mechanism, some works in the literature agree that GRUs and LSTMs provide a comparable performance in different machine learning tasks [29, 26, 30].

This work continues these efforts by further revising GRUs and proposing an architecture potentially more suitable for speech recognition. The contribution of this paper is twofold: First, we propose to remove the reset gate from the network design. Similarly to [31], we found that removing the reset gate does not affect the system performance, since we observed a certain redundancy in the role played by the update and reset gates. Second, we propose to replace hyperbolic tangent (tanh) with Rectified Linear Units (ReLU) activations in the state update equation. In the past, such a non-linearity has been avoided for RNNs due to the numerical instabilities caused by the unboundedness of ReLU activations. However, when coupling our ReLU-based GRU architecture with batch normalization [4], we did not experience such numerical issues. This allows us to take advantage of ReLU neurons, which have been proven effective at further alleviating the vanishing gradient problem as well as speeding up network training.

Our results, obtained on different tasks, input features, and noisy conditions show that, in our implementation, the revised architecture reduces the per-epoch training wall-clock time with more than 30%, while improving the recognition performance over all the experimental conditions considered in this work.

2. Revising GRUs

The GRU architecture is defined by the following equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{1a}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{1b}$$
$$\widetilde{h_t} = \tanh(W_h x_t + U_h(h_{t-1} r_t) + b_h), \tag{1c}$$
$$h_t = z_t h_{t-1} + (1 - z_t)\widetilde{h_t}. \tag{1d}$$

In particular, z_t and r_t are vectors corresponding to the update and reset gates respectively, while h_t represents the state vector for the current time frame t. The activations of both gates are element-wise logistic sigmoid functions $\sigma(\cdot)$, which constrain z_t and r_t to take values ranging from 0 and 1. The candidate state $\widetilde{h_t}$ is processed with a hyperbolic tangent. The network is fed by the current input vector x_t (e.g., a vector with speech features), while the parameters of the model are the matrices W_z, W_r, W_h (the feed-forward connections) and U_z, U_r, U_h (the recurrent weights). The architecture finally includes trainable bias vectors b_z, b_r and b_h, which are added before the non linearities.

As shown in Eq. 1d, the current state vector h_t is a linear interpolation between the previous activation h_{t-1} and the current candidate state $\widetilde{h_t}$. The weighting factors are set by the update gate z_t, that decides how much the units will update their activations. Note that such a linear interpolation, which is similar to that adopted for LSTMs [24], is the key component for learning long-term dependencies. For instance, if z_t is close to one, the previous state is kept unaltered and can remain unchanged for an arbitrary number of time steps. On the other hand, if z_t is close to zero, the network tends to favor the candidate state $\widetilde{h_t}$, which depends more heavily on the current input and on the closer hidden states. The candidate state $\widetilde{h_t}$, also depends on the reset gate r_t, which allows the model to delete the past memory by forgetting the previously computed states.

The model proposed in this paper is a revised version of the GRUs described above. The main changes concern reset gate and ReLU activations, as outlined in the next sub-sections.

2.1. Removing the reset gate

The reset gate can be useful when significant discontinuities occur in the sequence. For language modeling, this may happen when moving from one text to another one which is not-semantically related. In such situations, it is convenient to reset the stored memory in order to avoid taking a decision biased by an uncorrelated history. However, for some specific tasks such a functionality might not be useful. In [31], for instance, removing r_t from the GRU model has led to a single-gate architecture called Minimal Gated Recurrent Unit (M-GRU), which achieves a performance comparable to that obtained by standard GRUs in handwritten digit recognition as well as in a sentiment classification task.

We argue that the role played by the reset gate should be reconsidered also for acoustic modeling in speech recognition. In fact, a speech signal is a sequence that evolves rather slowly (the features are typically computed every 10 ms), in which the past history can virtually always be helpful. Even in the presence of strong discontinuities, for instance observable at the boundary between a vowel and a fricative, completely resetting the past memory can be harmful. On the other hand, it is helpful to memorize phonotactic features, since some phone transitions are more likely than others.

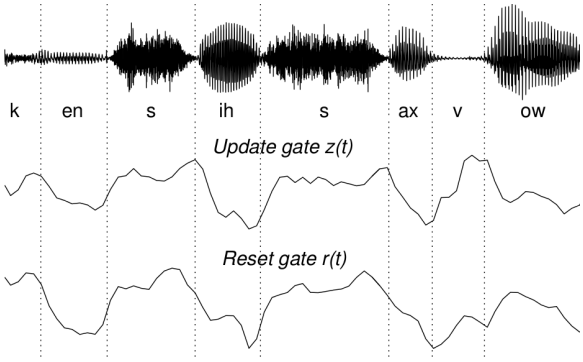

Figure 1: *Average activations of update and reset gates for a GRU trained on TIMIT in a chunk of the utterance "sx403" of speaker "faks0".*

Moreover, we believe that a certain redundancy in the activations of reset and update gates might occur when processing speech sequences. For instance, when it is necessary to give more importance to the current information, the GRU model can set small values of r_t. A similar effect can also be achieved with the update gate only, by setting small values of z_t. The latter solution tends to weight more the candidate state $\widetilde{h_t}$, which, as desired, depends more heavily on the current input and on its more recent history. Similarly, a high value can be assigned either to r_t or to z_t, in order to place more importance on past states. This redundancy is also highlighted in Fig. 1, where a temporal correlation in the average activations of update and reset gates can be readily appreciated for a GRU trained on TIMIT.

The first modification to standard GRUs proposed in this work thus concerns the removal of the reset gate r_t, which helps in limiting the redundancy in the gating mechanism. The main benefits of this intervention are related to the improved computational efficiency, which is achieved thanks to the reduced number of parameters necessary to reach the performance of a standard GRU.

2.2. ReLU activations

The second modification consists in replacing the standard hyperbolic tangent with ReLU activations in the state update equations (Eq. 1c - 1d). Tanh activations are, indeed, rather critical since their saturation slows down the training process and causes vanishing gradient issues. The adoption of ReLU-based neurons, which have shown effective in improving such limitations, was not so common in the past for RNNs, due to numerical instabilities originated by the unbounded ReLU functions applied over long time series. Nevertheless, some recent works have shown that ReLU RNNs can be effectively trained with a proper orthogonal initialization [32].

Formally, removing the reset gate and replacing the hyperbolic tangent function with the ReLU activation now leads to the following update equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{2a}$$
$$\widetilde{h_t} = \mathrm{ReLU}(W_h x_t + U_h h_{t-1} + b_h), \tag{2b}$$
$$h_t = z_t h_{t-1} + (1 - z_t)\widetilde{h_t}. \tag{2c}$$

We called this architecture M-reluGRU, to emphasize the modifications carried out on a standard GRU.

2.3. Batch Normalization

Batch normalization [4] has been recently proposed in the machine learning community and addresses the so-called *internal covariate shift* problem by normalizing, for each training mini-batch, the mean and the variance of each layer pre-activations. Such a technique has shown to be crucial both to improve the system performance and to speed-up the training procedure. Batch normalization can be applied to RNNs in different ways. In [33], authors suggest to apply it to feed-forward connections only, while in [34] the normalization step is extended to recurrent connections, using separate statistics for each time-step.

In this work, we tried both approaches and we observed a comparable performance between them. We also noticed that coupling the proposed model with batch-normalization [4] helps in avoiding the numerical issues that often occur when dealing with ReLU RNNs applied to long time sequences. Batch normalization, indeed, rescales the neuron pre-activations, inherently bounding the values of the ReLU neurons. This allows the network to take advantage of the well-known benefits of such activations.

3. Experimental setup

3.1. Corpora and tasks

To provide an accurate evaluation of the proposed architecture, the experimental validation was conducted using different training datasets, tasks and environmental conditions. A first set of experiments with TIMIT was performed to test the proposed model in a close-talking scenario. The experiments with TIMIT are based on the standard phoneme recognition task, which is aligned with that proposed in the Kaldi s5 recipe [35].

A set of experiments was also conducted in a distant-talking scenario with a Wall Street Journal (WSJ) task, to validate our model in a more realistic situation. The reference context was a domestic environment characterized by the presence of non-stationary noise (with an average SNR of about 10dB) and acoustic reverberation (with an average reverberation time T_{60} of about 0.7 seconds). The original WSJ training dataset was contaminated with a set of impulse responses measured in a real apartment. The test phase was carried out with the DIRHA-English corpus[1] (with both real and simulated datasets), consisting of 409 WSJ sentences uttered by six native American speakers in the same apartment. A development set of 310 WSJ sentences uttered by six different speakers was also used for hyperparameter tuning. More details on this dataset and on the impulse responses can be found in [36, 37].

3.2. System details

The architecture adopted for the experiments consisted of multiple recurrent layers, which were stacked together prior to the final softmax context-dependent classifier. These recurrent layers were bidirectional RNNs [11], which were obtained by concatenating the forward hidden states (collected by processing the sequence from the beginning to the end) with backward hidden states (gathered by scanning the speech in the reverse time order). Recurrent dropout was used as regularization technique. Since extending standard dropout to recurrent connections hinders learning long-term dependencies, we followed the approach introduced in [38], which tackles this issue by sharing the same dropout mask across all the time steps. Moreover, batch normalization was adopted exploiting the method suggested in [33], as discussed in Sec. 2. The feed-forward connections of the architecture were initialized according to the *Glorot* initialization [39], while recurrent weights were initialized with orthogonal initialization [32]. Similarly to [40], the gain factor γ of batch normalization was initialized to $\gamma = 0.1$ and the shift parameter β was initialized to 0.

Before training, the sentences were sorted in ascending order according to their lengths and, starting from the shortest utterances, minibatches of 8 sentences were progressively processed by the training algorithm. This sorting approach, besides minimizing zero-paddings when forming mini-batches, exploits a curriculum learning strategy [41] which has been shown to improve the performance and to ensure numerical stability of gradients. The optimization was done using the Adam algorithm [5], which ran for 22 epochs. The performance on the development set was monitored after each epoch, while the learning rate was halved when the performance improvement went below a certain threshold. Gradient truncation was not applied, allowing the system to learn arbitrarily long time dependences. The speech recognition labels were derived by performing a forced alignment procedure on the original training datasets. See the standard s5 recipe of Kaldi for more details [35].

To evaluate the proposed architecture on different input features, three sets of experiments were performed with 39 MFCCs, with 40 mel-filterbank coefficients, as well as with 40 fMLLR features derived by a Speaker Adaptive Training (SAT) procedure [35]. All of these feature vectors were computed every 10 ms with a frame length of 25 ms.

The main hyperparameters of the model (i.e., learning rate, number of hidden layers, hidden neurons per layer, dropout factor) were optimized on the development data. In particular, we guessed some initial values according to our experience, and starting from them we performed a grid search to progressively explore better configurations. A total of 20-25 experiments were conducted for the RNN models. As a result, an initial learning rate of 0.0013 and a dropout factor of 0.2 were chosen for all the experiments. The optimal numbers of hidden layers and hidden neurons, instead, depend on the considered dataset/model, and range from 4 to 5 hidden layers with 375-607 neurons.

The proposed system was implemented with Theano [42] and coupled with the Kaldi decoder [35] to form a context-dependent DNN/HMM speech recognizer[2].

4. Results

In the following sub-sections, a comparison of the proposed architecture with other popular RNNs is presented. The results will be reported for the standard close-talking TIMIT dataset and for the distant-talking WSJ task based on the DIRHA English dataset.

4.1. Results on TIMIT

Table 1 presents the results obtained with the TIMIT dataset. To perform a more accurate comparison of the various architectures, at least five experiments varying the initialization seeds were conducted for each RNN model. The results in Table 1 are reported as the average Phone Error Rates (PER%) with their corresponding standard deviations.

[1] This dataset is being distributed by the Linguistic Data Consortium (LDC).

[2] The code used for the experiments can be found at `https://github.com/mravanelli/theano-kaldi-rnn/`.

Arch. \ *Feat.*	MFCC	FBANK	fMLLR
relu-RNN	18.7 $\pm$ 0.18	18.3 $\pm$ 0.23	16.3 $\pm$ 0.11
LSTM	18.1 $\pm$ 0.33	17.1 $\pm$ 0.36	15.7 $\pm$ 0.32
GRU	17.1 $\pm$ 0.20	16.7 $\pm$ 0.36	15.3 $\pm$ 0.28
M-GRU	17.2 $\pm$ 0.11	16.7 $\pm$ 0.19	15.2 $\pm$ 0.10
M-reluGRU	**16.7** $\pm$ 0.26	**15.8** $\pm$ 0.10	**14.9** $\pm$ 0.27

Table 1: *Phone Error Rate (PER%) obtained for the test set of TIMIT with various RNN architectures.*

Architecture	Layers	Neurons	# Params	Training time
relu-RNN	4	607	6.1 M	318 sec.
LSTM	5	375	8.8 M	506 sec.
GRU	5	465	10.3 M	580 sec.
M-GRU	5	465	7.4 M	390 sec.
M-reluGRU	5	465	7.4 M	390 sec.

Table 2: *Comparison of the per-epoch training time for the RNN architectures optimized on the TIMIT development set.*

The first row of Table 1 presents the results of a traditional RNN with ReLU activations (no gating mechanisms are used here). Although this architecture has recently shown promising results in some machine learning tasks [32], our speech recognition performance results confirm that gated recurrent networks (rows 2-5) still outperform traditional RNNs. We also observed that GRUs tend to slightly outperform the LSTM, in the tasks addressed in these experiments.

The M-GRU architecture is the version of GRU without the reset gate. Table 1 highlights that the M-GRU achieves a performance very similar to that obtained with standard GRUs, further confirming our speculation on the negligible role played by the reset gate in a speech recognition application. The last row of Table 1 reports the performance achieved with the proposed model, in which, besides removing the reset gate, ReLU activations are used. Results indicate that M-reluGRU consistently achieves the best performance across the various input features. A remarkable achievement is the average PER(%) of 14.9% obtained with the proposed architecture using fMLLR features. To the best of our knowledge, such a result yields the best published performance on the TIMIT test-set.

Table 2 reports a comparison of the per-epoch training time for all the considered RNNs. Results confirm that the main advantage of the proposed model is its ability to significantly reduce the training time. In our Theano implementation, running each GRU epoch lasts 580 seconds on an NVIDIA K40 GPU against just 390 seconds taken by the M-reluGRU, with a training time reduction of more than 32%. Table 2 also reports the best architectures obtained after optimizing the hyperparameters on the TIMIT development set. Results show that for GRU models the best performance is achieved with 5 hidden layers of 465 neurons.

4.2. Results on DIRHA English WSJ

Tables 3 and 4 summarize the results obtained with the simulated and real parts of the DIRHA English WSJ dataset. For the sake of compactness, only the average performance (obtained by running five experiments with different initialization seeds) is reported, while standard deviations (which range from 0.2% to 0.4%) are omitted.

Arch. \ *Feat.*	MFCC	FBANK	fMLLR
relu-RNN	23.7	23.5	18.9
LSTM	23.2	23.2	18.9
GRU	22.3	22.5	18.6
M-GRU	21.5	22.0	18.0
M-reluGRU	**21.3**	**21.4**	**17.6**

Table 3: *Word Error Rate (%) obtained with the DIRHA English WSJ dataset (simulated part) for various RNN architectures.*

Arch. \ *Feat.*	MFCC	FBANK	fMLLR
relu-RNN	29.7	30.0	24.7
LSTM	29.5	29.1	24.6
GRU	28.5	28.4	24.0
M-GRU	28.4	28.1	23.6
M-reluGRU	**27.8**	**27.6**	**22.8**

Table 4: *Word Error Rate (%) obtained with the DIRHA English WSJ dataset (real part) for various RNN architectures.*

Table 3 and 4 exhibit a trend comparable to that observed for the TIMIT dataset, confirming the effectiveness of the proposed architecture also in a more realistic and challenging scenario. The results are consistent over both real and simulated data as well as across the different features considered in this study. The reduction of the training time is about 36% (25 minutes for per-epoch training with the proposed model against 40 minutes spent by the standard GRU).

Moreover, the reset gate removal seems to play a more crucial role in the addressed distant-talking scenario. If the close-talking results reported in Table 1 highlight comparable error rates between standard GRU and M-GRU, in the distant-talking case we even observe a performance gain when removing the reset gate. We suppose that this behaviour is due to reverberation, which implicitly introduces redundancy in the signal, due to the multiple delayed replicas of each sample. This results in a forward memory effect that smooths energy envelopes as well as sub-band energy contours. Due to this effect, a reset gate mechanism might become ineffective to forget the past.

5. Conclusions

In this paper, we revised standard GRUs for speech recognition purposes. The proposed architecture is a simplified version of a GRU, in which the reset gate is removed and ReLU activations are used. The experiments, conducted on different tasks, features and environmental conditions, have confirmed the effectiveness of the proposed model not only to reduce the computational complexity of our implementation (with a reduction of more than 30% of the training time over a standard GRU), but also to slightly improve the recognition performance.

Future efforts will be focused on extending this work to different tasks and larger datasets (e.g., switchboard or LibriSpeech) and to test the revised architecture on modern end-to-end systems (such as CTC and attention-based models). To ensure the practical value of the computational gains of our models, these experiments should be replicated using more optimized implementations.

6. References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] D. Yu and L. Deng, *Automatic Speech Recognition - A Deep Learning Approach*. Springer, 2015.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of ICML*, 2015, pp. 448–456.

[5] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.

[6] H. Bourlard and N. Morgan, "Continuous speech recognition by connectionist statistical methods," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 893–909, 1993.

[7] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[8] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, Oct 2014.

[9] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[10] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc of INTERSPEECH*, 2015, pp. 3214–3218.

[11] A. Graves, N. Jaitly, and A. Mohamed., "Hybrid speech recognition with Deep Bidirectional LSTM," in *Proc of ASRU*, 2013.

[12] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. of INTERSPEECH*, 2014, pp. 338–342.

[13] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," 2015. [Online]. Available: http://arxiv.org/abs/1512.02595

[14] Z. Chen, S. Watanabe, H. Erdogan, and J. Hershey, "Speech enhancement and recognition using multi-task learning of long short-term memory recurrent neural networks," in *Proc. of INTERSPEECH*, 2015, pp. 3274–3278.

[15] T. Menne, J. Heymann, A. Alexandridis, K. Irie, A. Zeyer, M. Kitza, P. Golik, I. Kulikov, L. Drude, R. Schlüter, H. Ney, R. Haeb-Umbach, and A. Mouchtaris, "The RWTH/UPB/FORTH System Combination for the 4th CHiME Challenge Evaluation," in *CHiME 4 challenge*, 2016, pp. 39–44.

[16] F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. L. Roux, J. R. Hershey, and B. W. Schuller, "Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR," in *Proc. of LVA/ICA*, 2015, pp. 91–99.

[17] F. Weninger, J. Le Roux, J. Hershey, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *Proc. of IEEE GlobalSIP*, 2014.

[18] H. Erdogan, J. R. Hershey, S. Watanabe, and J. L. Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proc. of ICASSP*, 2015, pp. 708–712.

[19] F. Eyben, F. Weninger, S. Squartini, and B. Schuller, "Real-life voice activity detection with lstm recurrent neural networks and an application to hollywood movies," in *Proc. of ICASSP*, 2013, pp. 483–487.

[20] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. of ICML*, 2014, pp. 1764–1772.

[21] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-End Attention-based Large Vocabulary Speech Recognition," in *Proc. of ICASSP*, 2016.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transaction on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[23] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks." in *Proc. of ICML*, 2013, pp. 1310–1318.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[25] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–11, 2016.

[26] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. of ICML*, 2015, pp. 2342–2350.

[27] Y. Zhang, G. Chen, D. Yu, K. Yao, S. Khudanpur, and J. R. Glass, "Highway long short-term memory RNNS for distant speech recognition," in *Proc. of ICASSP 2016*, 2016, pp. 5755–5759.

[28] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proc. of SSST*, 2014.

[29] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. of NIPS*, 2014.

[30] K. Irie, Z. Túske, T. Alkhouli, R. Schlúter, and H. Ney, "LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition," in *Proc. of INTERSPEECH*, 2016.

[31] G. Zhou, J. Wu, C. Zhang, and Z. Zhou, "Minimal gated unit for recurrent neural networks," 2016. [Online]. Available: http://arxiv.org/abs/1603.09420

[32] Q. Le, N. Jaitly, and G. Hinton, "A simple way to initialize recurrent networks of rectified linear units," 2015. [Online]. Available: http://arxiv.org/abs/1504.00941

[33] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *Proc. of ICASSP*, 2016, pp. 2657–2661.

[34] T. Cooijmans, N. Ballas, C. Laurent, and A. Courville, "Recurrent batch normalization," 2016. [Online]. Available: http://arxiv.org/abs/1603.09025

[35] D. Povey *et al.*, "The Kaldi Speech Recognition Toolkit," in *Proc. of ASRU*, 2011.

[36] M. Ravanelli, L. Cristoforetti, R. Gretter, M. Pellin, A. Sosi, and M. Omologo, "The DIRHA-English corpus and related tasks for distant-speech recognition in domestic environments," in *Proc. of ASRU*, 2015, pp. 275–282.

[37] M. Ravanelli, P. Svaizer, and M. Omologo, "Realistic multi-microphone data simulation for distant speech recognition," in *Proc. of INTERSPEECH*, 2016.

[38] T. Moon, H. Choi, H. Lee, and I. Song, "RNNDROP: A novel dropout for RNNS in ASR," in *Proc. of ASRU*, 2015, pp. 65–70.

[39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of AISTATS*, 2010, pp. 249–256.

[40] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Batch-normalized joint training for dnn-based distant speech recognition," in *Proc. of SLT*, 2016.

[41] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. of ICML*, 2009, pp. 41–48.

[42] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.